An MBASIC Application Program for Relational Inquiries on a Database

R. M. Smith
DSN Facility Operations Office

An MBASIC application program is described that allows a user to specify and perform a sequence of relational operations on a database.

I. Introduction

A previous article (Ref. 1) dealt with the recent implementation of a relational database in MBASIC, and focused upon the realization of the use of the five relational operations (Refs. 2 and 3).

This article describes an MBASIC application program that allows a user to specify and use a sequence of relational operations on a relational database for the purpose of making an inquiry or for the purpose of transferring data to a new file.

II. Relational Structure

In the relational model of data, data are organized into arrays (called relations) with fields (called domains), so that each record entry (called a tuple) is a set of attributes describing the characteristics of a real world entity. For instance the assignment of test equipment might be represented by the following notation:

'ASSIGNMENT' (CON,OWNER,LOCATION,OPSTAT, REC DATE)

where:

CON = control number

OWNER = facility responsible for the equipment

LOCATION = facility where equipment is presently located

OPSTAT = operational status

REC DATE = date received at the indicated location

In the same manner, the failure of an item of test equipment might be represented:

'FAILURE' (CON,DATE,TIME,DOWN#,FACILITY)

where:

DATE and TIME = date and time of the failure occurrence

DOWN# = a unique event number for the failure occurrence

FACILITY = the operational facility at which the failure occurred

Finally, the status of a subsystem (containing test equipment) might be represented:

'SSDOWN' (DOWN#,SSMA,UNIT,REFDES,DATE, TIME)

where:

SSMA = subsystem-major assembly number

UNIT = equipment rack unit number where the failure occurred

REFDES = reference designator (a coordinate position within a subsystem unit) where the failure occurred.

DATE and TIME = date and time that the subsystem became inoperative

Other relations may be constructed to describe other relationships but these three will be used as illustrations in this paper.

III. Relational Operations

There are five operations specified in the relational model and explained in Refs. 1, 2, and 3:

(1) Join

(2) Restriction

(3) Division

(4) Projection

(5) Permutation

Specific notation is used as a convenient way of expressing relational operations. The notation is simple and allows the annotation of a database access strategy and can be converted easily to MBASIC code or used with the application program discussed in this article.

Typical notation is as follows:

(1) Restriction

'RELATION' | DOMAIN1 = selected data value

stated: restrict 'RELATION' on DOMAIN1 equal to selected data value

meaning: select from 'RELATION' all tuples where the data value in DOMAIN is equal to the selected data value.

(2) Join

'RELATION1' (DOMAIN1) 'RELATION2'

stated: join 'RELATION1' with 'RELATION2' over DOMAIN1

meaning: compare tuples in 'RELATION1' with tuples in 'RELATION2' and, where the data value in DOMAIN1 is the same in both tuples, produce a resultant tuple that contains the domains of both tuples.

(3) Projection

π 'RELATION3' (DOMAIN1, DOMAIN2, . . . DOMAINn)

stated: project 'RELATION3'

meaning: create 'RELATION3' containing DOMAIN1, DOMAIN2, . . . DOMAINn

(4) Permutation

Permutation is a function of the projection operation where the resultant domain order is changed from the original order:

π 'RELATION4' (DOMAIN2, DOMAIN6, $\cdots$, DOMAINn)

(5) Division

Division is a special case of restriction and projection in this implementation and will not be dealt with separately.

IV. Description of the Application Program

'INQUIRY' is a generalized program allowing a user to use the relational operations singly or in a selected sequence to manipulate the domains of one or more relations in a database. The program is data independent; that is, it does not need to be rewritten or altered if the database is reorganized. This feature is realized by the use of a directory relation that contains a description of the database. The program derives the current data description from the directory relation. When 'INQUIRY' is used to create a new relation, the directory relation is automatically updated to contain the new description. When a "temporary" relation is created the program will delete the directory update at the request of the user.

'INQUIRY' accepts relational operations in the following formats from an operator:

(1) Restriction

RELATION1, DOMAIN1, =, data value[1]

(2) Join

RELATION1, DOMAIN1, RELATION2

(3) Projection

RELATION1, QD, RELATION3, DOMAIN2, • • •, DOMAINn

where RELATION1 and RELATION2 are existing relations; QD is the quantity of domains to be projected; RELATION3 is a new relation to be created (may be the terminal); DOMAIN1 is an existing domain; DOMAIN2 through DOMAINn are the domains to be contained in the new relation.

Compare these formats with the standard notation formats in Section III of the article.

The relational operations may be used in the following combinations in 'INQUIRY':

(1) Restriction

(2) Projection

(3) Join

(4) Restriction, join

(5) Restriction, projection

(6) Restriction, join, projection

(7) Join, projection

(8) Join, restriction

(9) Join, restriction, projection

Projection: may be to:

(1) Terminal ("TERM")

(2) Permanent relation (NAME)

(3) Temporary relation (*NAME)

Figure 1 is a functional flowchart of the program and illustrates the possible combinations graphically.

V. Applications

'INQUIRY' may be used as a general purpose access program or as a design tool in planning the strategy of a single purpose program.

Figure 4 illustrates a session at a terminal using 'INQUIRY' to extract data from the relation called 'ASSIGNMENT' (Fig. 5). The process depicted in Fig. 4 is a restriction followed by a projection, the same process that is depicted in Fig. 2. Figure 6 illustrates a session at a terminal using 'INQUIRY' to manipulate data from two relations and to display a resultant data list at the terminal. The process depicted in Fig. 6 is a join followed by a projection, the same process that is depicted in Fig. 3. This sort of activity is representative of the general purpose aspect of the program.

To use 'INQUIRY' as a tool to design a single purpose program the user would "construct" using 'INQUIRY', a sequence of relational operations in a manner similar to that depicted in Fig. 6. If the resultant content and domain order were as desired then the user would convert the relational statements to MBASIC code using additional code as necessary for column headings, spacing, titling, etc.

Figures 2 and 3 depict, respectively, the restriction and join operations first in standard relational notation then in typical flowchart configuration and finally in MBASIC code. The applications described in this article require the use of relations that are in, at least, first normal form (Ref. 3). 'ASSIGNMENT' (Fig. 5), which is used in some of the examples in this article, is in third normal form (Ref. 3).

[1]In the restriction operation, "=", ">", or "<" may be used.

References

1. Smith, R. M., "A Relational Data Base Implemented Using MBASIC," in *The Deep Space Network Progress Report 42-30*, pp. 291-305, Jet Propulsion Laboratory, Pasadena, Calif., Dec. 15, 1975.

2. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," *Communications of ACM*, Vol. 13, No. 6, June 1970.

3. Date, C. J., *An Introduction to Database Systems*, Addison-Wesley Publishing Co., Inc., Reading, Mass., 1975.

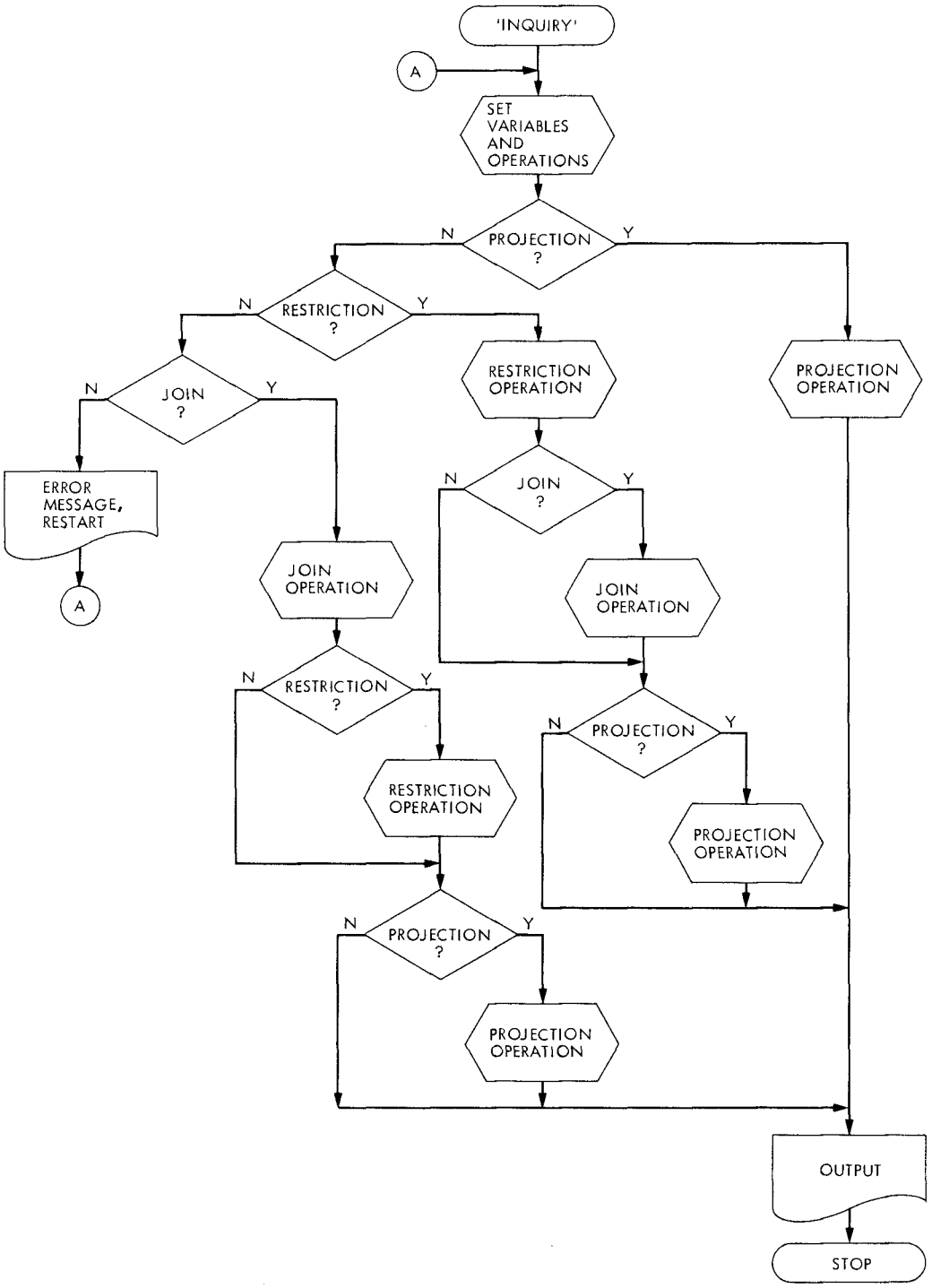
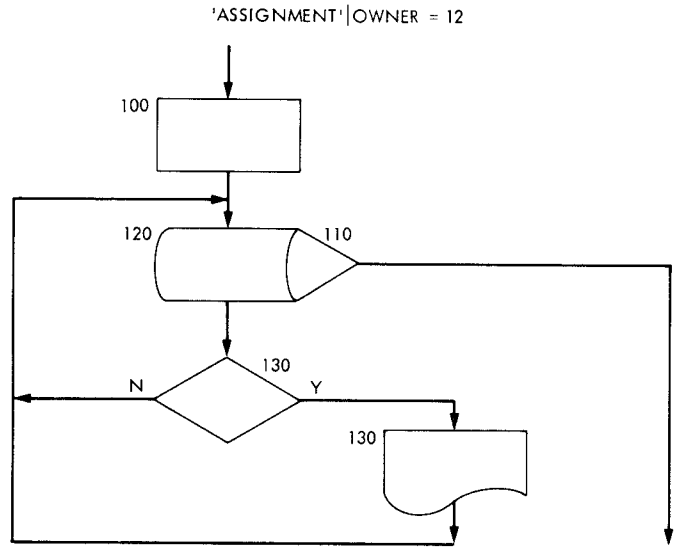


Fig. 1. Functional flowchart for 'INQUIRY'

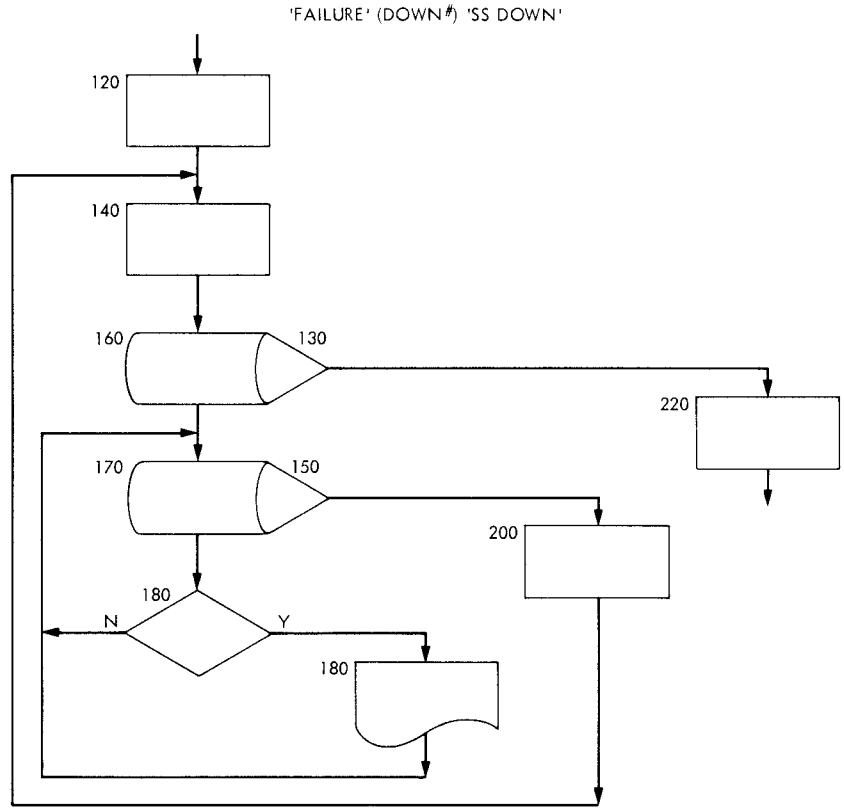



```
100 OPEN 'ASSIGNMENT',INPUT,1
110 AT ENDFILE(1) GO TO 150

120 INPUT FROM 1:C$(1),O$(1),L$(1),OS$(1),PD$(1)
130 IF O$(1)='12' THEN PRINT C$(1);L$(1);OS$(1)
140 GO TO 120

150 CLOSE 1
```

Fig. 2. The restriction operation




```
120 OPEN 'FAILURE',INPUT,1
130 AT ENDFILE(1) GO TO 220

140 OPEN 'SSDOWN',INPUT,2
150 AT ENDFILE(2) GO TO 200
160 INPUT FROM 1:C$(1),DI1$(1),TD$(1),DN1$(1),L$(1)
170 INPUT FROM 2:DN2$(1),SS$(1),UN$(1),REF$(1),DI2$(1),TI2$(1)
180 IF DN1$(1)=DN2$(1) THEN PRINT L$(1);SS$(1);DI1$(1);DN1$(1)
190 GO TO 170
200 CLOSE 2
210 GO TO 140

220 CLOSE 1,2
```

Fig. 3. The join operation

```
>RUN
?R
?ASSIGNMENT.OWNER.=.12
JOIN? N
PROJECTION? Y
?3,TERM,CON,LOCATION,OPSTAT

BB5C11  12 DL
CR6B12  1Y US

***END
```

Fig. 4. Using 'INQUIRY' to plan the restriction strategy of Fig. 2

```
COPY 'ASSIGNMENT' TO TERMINAL
AA3A12,1Y,1Y,DL,010675
AB6C21,1X,1X,DL,170273
CD3B15,1X,1X,SP,121072
BB5C12,14,14,DL,091274
AX3B09,11,1Y,EP,190176
DA4C12,1X,1X,DL,071071
BB5C11,12,12,DL,091274
AR7D15,1Y,1X,US,250276
BB5C13,11,11,SP,091274
CC7C02,1X,1X,US,151172
CX5B13,1Y,1Y,DL,151071
CR6B12,12,1Y,US,110376
>
```

Fig. 5. A listing of data contained in the relation 'ASSIGNMENT'

```
RUN
?J
?RMS*EQPT.FAILURE,DOWN#,RMS*EQPT.SSDOWN
RESTRICTION? N
PROJECTION ?Y
?4,TERM,FACILITY,SSMA,DATE,DOWN#
12  3606  100376  3775

DO *FILES NEED TO BE DELETED? N

***END
>
```

Fig. 6. Using 'INQUIRY' to plan the join strategy of Fig. 3